An algebra for structured documents in the context of the object-oriented approach

Sylviane R. Schwer, Haider Hamza, André Flory

An algebra for structured documents in the context of the object-oriented approach

Haider Hamza
INSA, LISI, 20, Avenue Albert Einstein 69621 Villeurbanne France
e-mail : hamza@lisiflory.insa-lyon.fr

Sylviane R. Schwer
Université Paris 13, LIPN, av. J. B.. Clément, 93430 Villetaneuse, France
e-mail:schwer@lipn.univ-paris13.fr

André Flory
INSA, LISI, 20, Avenue Albert Einstein 69621 Villeurbanne France
e-mail : flory@lisiflory.insa-lyon.fr

Summary

The aim of this paper consists in defining an algebra allowing the request of structured documents in the context of object oriented approach. The operators of this algebra are defined in conformity with the concepts of the object model. It takes into consideration not only the embedding of structures and links between them, but also the informational aspect of the document as well as its structural aspect.

The operands of the proposed algebra are only one type: they are subdatabases. A subdatabase is made of a collection of the databases objects, grouped in classes and interconnected through links. This algebra responds to two objectives. Firstly, it represents a basic nucleus of a declarative query system; it contains the whole of the elementary operations which will be used in the resolution of the request. Secondly, it provides facilities to users to formulate their requests and manipulate the documentary database.

Key Words : Structured documents, Object oriented, Documentary database, Query, Algebra

1. Introduction

Since the emergence of the relational model, many researches have addressed the problem of the development of documentary databases using the technology of database management relational systems (RDBMS). For the development of documentary applications, the interest of this approach lies in the possibility of taking advantage of all the functionality of RDBMSs. Nevertheless the capacities of abstraction and the expressiveness of the relational model are insufficient for handling structured documents. Indeed, the representation of the fragments of the documents by several tables results in a heavy modeling and decreases considerably the performance of the system.

Researches aiming at a new generation of DBMS has been carried out to extend the models of data representation and to increase the power of manipulation languages. Indeed, new needs have emerged requiring references to the stocked information in terms of the perception of the user and not simply according to their model inside the database . The object oriented DBMSs gives nowadays a quite rich model capable of representing complex data. A document can be modeled by a complex object the components of which may be formed by other objects belonging to different classes. The classes are organized in a hierarchy of inheritance in which the objects of different classes do respect the relations of inclusion between them. The technology of OODBMS is well adapted to the management of structured documents such as arborescences of objects with associated methods [Amgh89, Bens89, Moua89, Hamz96].

The retrieval of informations is one of the most important functionality of a management system of documentary databases. By system of retrieval of information, we mean the whole mechanisms which allows the user to select documentary information. One of the essential objectives of a retrieval system of informations is to make easy the restitution of a portion of information from a documentary database, in response to a user's request. Different strategies of information retrieval were developed, and most of them were built around an algebra. In fact, G□ting [Guti89] has proposed an algebra made of a set of operators which permits to take into account the different informational aspects of a document. But this algebra is not well adapted to an object-oriented system, because it doesn't consider the inter-objects relationships. In the case of the algebra for the OODBMSs, several works were carried out [Alha93, Clue90, Hamz95, Liu93, Stan93, Shaw90, Su93, Subr95]. However, these algebra are not really adapted to structured documents, since they only take into consideration the informational aspect of the data, and not at all their structural aspect. This is the case, for instance, of the algebra of Show and Zdonik, which is an extension

of the algebra of the denormalized models that takes into account some object-oriented concepts. The EXCESS algebra [vand91], is a many-sort algebra i.e. the operators are defined by types of operands. Such an approach in a documentary retrieval context puts in question the uniformity of the operators and the reusing of the results of the requests for a new query. The algebra of Liu [Liu93] is based on a clear distinction between links of association and links of aggregation. We think that in fact this distinction is not relevant: two conceptors can have different visions of the same reality. While one can model a relation as a link of composition, most models among the reviewed aboved treat it as a link of association.

The objective of our work consists in defining an algebra that allows the interrogation of documents in the context of an object-oriented approach. The basis of manipulation is the embedding of structures and the links between structures [Schw97]. This algebra permits to take into account the informational aspect as well as the structural aspect of documentary databases.

This paper is divided into 5 sections. The first one being this introduction, section 2 is devoted to the basic concepts of the proposed algebra. Section 3 is dedicated to the description of the algebraic operators and section 4 to examples of the algebraic operations. The last section consists in our conclusion and perspectives.

2. Formalization of a documentary database

In this part we define the basis of our algebra. Starting from a clean definition of what is a documentary database, we defined what is a subdatabase, the single type of our algebra and their operands which fit to the object-oriented ontology.

2.1. Documentary database

As usual in database theory, a documentary database is defined y a schema and an instance of this schema.

Definition 1: Documentary database schema

The documentary database schema can be defined by a finite acyclic connect graph. Formally, a schema Σ of documentary database is defined by the triplet (C, ρ, λ) where:

- $C = \{c_i\}$ is a set of classes, where each class c_i is characterized by some properties $(p_{i1}, p_{i2}, \ldots, p_{in})$.
- $\rho : C \longrightarrow 2^C$ is an hierarchy function which associates a set of classes to a given class. We note by ρ^{-1} the inverse relation of the hierarchy function, ρ^n the transitive hierarchy function of order n, ρ^* the transitive hierarchy function and ρ^+ the strict transitive hierarchy function which are defined as follows:
  - $\rho^0(c) = c$
  - $\rho^n(c) = \rho(\rho^{n-i}(x))$
  - $\rho^*(c) = \bigcup_{n \geq 0} \rho^n$
  - $\rho^+(c) = \bigcup_{n > 0} \rho^n$
- $\lambda : C \times C \longrightarrow 2^R$ is a Roll function which associates a set of rolls to a pair of classes. The schema is endowed by the following properties :

1) $Root(\Sigma) = \{c \in C / \forall c' \in C, \quad c' \notin \rho(c)\} \neq \varnothing$ (due to the fact that the graph is acyclic and finite)
2) $(\forall c \in C), \quad (c \notin \rho^+(c))$ (due to acyclism)
3) $(\forall c \notin Root(\Sigma)), \quad (\rho^{-1}(c) \neq \varnothing)$ (this is the definition of the Root)
4) $(\forall c, c' \in Root(\Sigma)), \quad (\rho^*(c) \cap \rho^*(c') \neq \varnothing)$ (this is due to the connection property)

Example:
Let $\Sigma = (C, \rho, \lambda)$ be the schema of the documentary database represented by fig. 1. The graph which corresponds to this schema is represented by fig. 2.

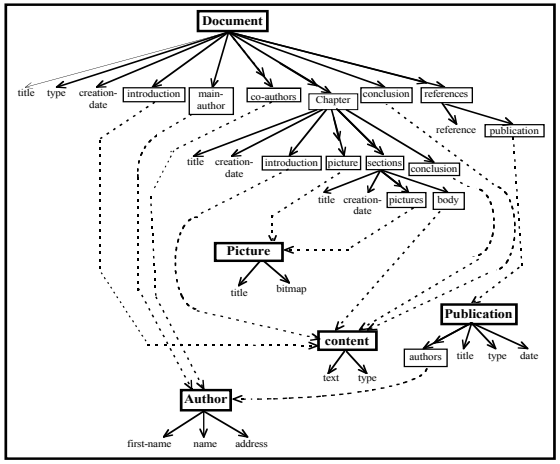

*Fig. 1. Example of a **schema** of a documentary database*

C={Document (title, type, date-creation), Chapter (title, date-creation), Reference (reference), Section (title, date-creation), Content (text, type), Image (title, bitmap), Author (nom, name, address)}

ρ(Document) = { Author, Content, Chapter, Reference }
ρ(Chapter) = { Content, Section, Picture }
ρ(Section) = { Content, Picture }
ρ(Reference) = { Publication }
ρ(Publication) ={ Author (authors) }
ρ(Content) = ∅
ρ(Picture) = ∅
ρ(Author) = ∅

λ(Document, Author) = {main-author, co-authors}
λ(Document, Content) = {introduction, conclusion}
λ(Chapter, content) = {introduction, conclusion}
λ(Section, content) = {body}
λ(Section, Picture) = {pictures}
λ(Reference, Publication) = {publication}
λ(Chapter, Picture) = {pictures}
λ(Publication, Author) = {authors}

The documentary database schema can also be defined by a finite acyclic connected graph. We denoted it a *Σ – graph*

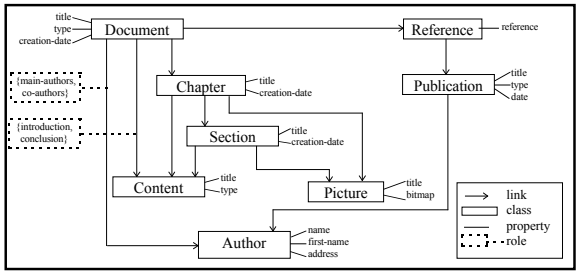

*Fig. 2. Representation of the **schema** of the documentary database under the form of a **connect acyclic graph***

A documentary database is defined by a schema and by an instance of this schema.

Definition 2 : Documentary database

A documentary database is defined by. (Σ, O, A) where $\Sigma = (C, \rho, \lambda)$ is a schema of the database, O is a set of objects and A a set of inter-objects links $A \subseteq O \times O$. If we note by τ the total function ($\tau : O \longrightarrow C$) which returns for every object its class, then (Σ, O, A) must verify the following constraint:

- $\forall (o_i, o_j) \in A,\ \tau(o_j) \in \rho(\tau(o_i))$

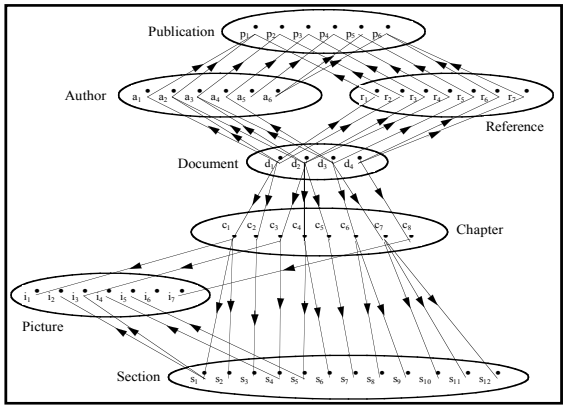

*Fig. 3. Example of a partiel view of an **instance** of a documentary database*

2.2. Documentary subdatabase

A user who wants to formulate queries on a database, is rarely interested by the whole entities in the database. In most cases, he wants to formulate queries on a portion of this database. For example, a user would like to formulate queries only on the *Author* class and *Document* class of our illustration.

We defined a subdatabase of a database D as a database which is merged into the database D.

Definition 3 : Subschema

A subschema $\psi = (C_\psi, \rho_\psi, \lambda_\psi)$ of a schema $\Sigma = (C_\Sigma, \rho_\Sigma, \lambda_\Sigma)$ is included in the Σ graph, such that :

- $C_\psi \subseteq C_\Sigma$
- $\forall c \in C_\psi \quad \rho_\psi(c) \subseteq \rho_\Sigma(c)$
- $\forall c_1, c_2 \in C_\psi \quad \lambda_\psi(c_1, c_2) \subseteq \lambda_\Sigma(c_1, c_2)$

Remarks:

- a schema is a subschema of itself.
- a class of a schema is a subschema of the schema.
- $\emptyset = (C_\emptyset, \rho_\emptyset, \lambda_\emptyset)$ ($C_\emptyset = \emptyset, \rho_\emptyset = \emptyset, \lambda_\emptyset = \emptyset$) is a subschema of any schema.

Example:

Fig. 4 represents the graph of the subschema $\psi = (C_\psi, \rho_\psi, \lambda_\psi)$ which is defined as:

C_ψ = { Document (title, type, creation-date), Chapter (title, creation-date), Reference (reference) }

ρ_ψ (Document) = {Chapter, Reference}

ρ_ψ (Chapter) = $\emptyset$

ρ_ψ (Reference) = $\emptyset$

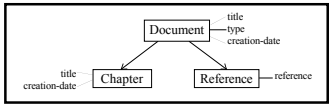

*Fig. 4. : graphical representation of a **subschema***

Definition 4 : An instance of a subschema

An instance of a subschema $\psi = (C_\psi, \rho_\psi, \lambda_\psi)$ is a set of objects and a set of links between objects which respectively correspond to the classes in the subschema and to the links between classes. Formally, an instance is defined by the pair:

$$(O, A)$$

where O is a set of objects and A a set of inter-objects links $A \subset O \times O$ with the following constraint:

- $\forall (x, y) \in A, \ \tau(y) \in \rho(\tau(x))$

Example:

Fig. 5 gives a graphical representations of an instance of the subschema that is represented in fig. 4.

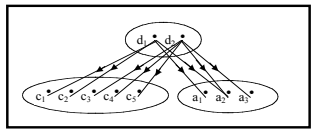

*Fig. 5 : An example of an **instance** of a subschema*

Definition 5 : Subdatabase

A subdatabase is defined by a subschema and by an instance of this subschema. Formally, a subdatabase is defined by the triple:

$$(\psi, O, A)$$

Example :

An example of a subdatabase is shown in fig. 6.

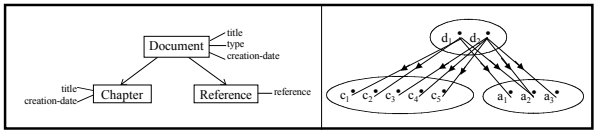

*Fig. 6 : A graphical representation of a **subdatabase**.*

Definition 6: path between two classes

Given a schema $\Sigma = (C_\Sigma, \rho_\Sigma, \lambda_\Sigma)$ and two classes *a, b* of C_Σ, a path *P(a, b),* is defined as a sequence $a.c_1.c_2.\ldots.c_n.b \ (n \geq 1)$ where:

- $c_i \in C_\Sigma$
- $c_i \in \rho(c_{i-1})$ or $c_{i-1} \in \rho(c) \quad 1 \leq i \leq n$

We can define alternately a path as a subschema $P(a, b) = (C_P, \rho_P)$ where:

- $C_P = \{a, b, c_1, \ldots, c_n\}$
- $\forall c_i \in C_P, \quad \rho_P(c_i) = \{c_{i+1}\}$ or $\rho_P^{-1}(c_i) = \{c_{i+1}\}$ depending of Σ

The number of classes a long path *P* is equal to *Length(P)= Card(*C_P*)*. The path length indicates the number of classes, including *a* and *b*, that must be traversed to reach the class *b*. Two classes can have several paths between them as of (*Document, Reference*).

Example :

P(Document, Reference) = {P_1*,* P_2*}*

P_1 = Document.Reference — $Len(P_1) = 2$

P_2 = Document.Author.Publication.Reference — $Len(P_2) = 4$

P_1 *can be defined as a subschema:* $P_1 = (\{Document, Reference\}, \{\rho(Document) = \{Référence\}, \lambda_\emptyset)$

Definition 7 : Set of paths between two subschemas

Given a schema $\Sigma = (C_\Sigma, \rho_\Sigma, \lambda_\Sigma)$, and two subschemas $\psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1})$ and $\psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2})$, the set of paths between ψ_1 and ψ_2 denoted by *SP(*ψ_1, ψ_2*),* is the set of all paths between ψ_1 and ψ_2. It is defined as follows:

$$SP(\psi_1, \psi_2) = \{P(c_i, c_j) / (c_i \in C_{\psi_1} \text{ and } c_j \in C_{\psi_2})\}$$

Num(SP(ψ_1, ψ_2*))* denotes the number of paths that exists between two subschemas ψ_1 and ψ_2.

Example:

The following is a set of paths for the subschemas ψ_1 and ψ_2 shown in fig. 6:

SP(ψ_1, ψ_2*) = {Document.Reference, Document.Author.Publication}*

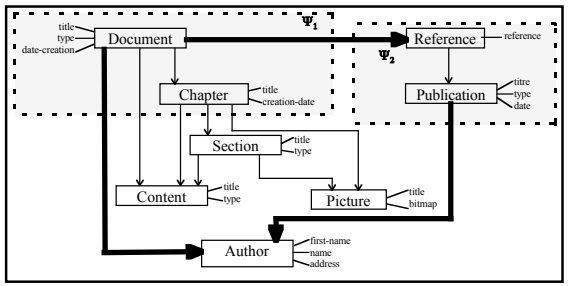

*Fig. 7. Representation of two subschemas and **paths** between them*

Definition 8: Distance between two subschemas

Given two subschemas $\psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1})$ and $\psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2})$, the distance between ψ_1 and ψ_2, denoted by $D(\psi_1, \psi_2)$, is defined as :

$$D(\psi_1, \psi_2) = Min\{Len(P) / P \in SP(\psi_1, \psi_2)\}$$

Example:

The following is a distance between subschemas ψ_1 and ψ_2 shown in figure 6:

$D(\psi_1, \psi_2)$ = *Min {Len(Document.Reference), Len (Document.Author.Publication)}* = 2

Definition 9: The upper limit of two subschemas

Given two subschemas $\psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1})$ and $\psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2})$, the upper limit $\psi_1 \vee \psi_2$ is the least subschema containing ψ_1 and ψ_2. The upper limit can be defined as the union of ψ_1, ψ_2 and all the paths between ψ_1 and ψ_2 that have their lengths equal to $D(\psi_1, \psi_2)$. Let us consider the set S of all the paths between ψ_1 and ψ_2 that have their lengths equal to $D(\psi_1, \psi_2)$. The set S is defined as :

$$S = \{(C_P, \rho_P, \lambda_\varnothing) / (C_P, \rho_P, \lambda_\varnothing) \in SP(\psi_1, \psi_2) : Len((C_P, \rho_P, \lambda_\varnothing)) = D(\psi_1, \psi_2)\}$$

Formally, the upper limit of two subschemas $\psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1})$ and $\psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2})$, is defined as:

$\psi_3 = \psi_1 \vee \psi_2 = (C_{\psi_3}, \rho_{\psi_3}, \lambda_{\psi_3})$ where :

- $C_{\psi_3} = C_{\psi_1} \cup C_{\psi_2} \bigcup_{i=1}^{n} C_{P_i}$
- $\forall c \in C_{\psi_3}, \quad \rho_{\psi_3}(c) = \rho_{\psi_1}(c) \cup \rho_{\psi_2}(c) \bigcup_{i=1}^{n} \rho_{P_i}(c)$
- $\forall c_1, c_2 \in C_{\psi_3} \quad \lambda_{\psi_3}(c_1, c_2) = \lambda_{\psi_1}(c_1, c_2) \cup \lambda_{\psi_2}(c_1, c_2)$

Example :

An example of an upper limit of two subschemas is shown in fig. 8.

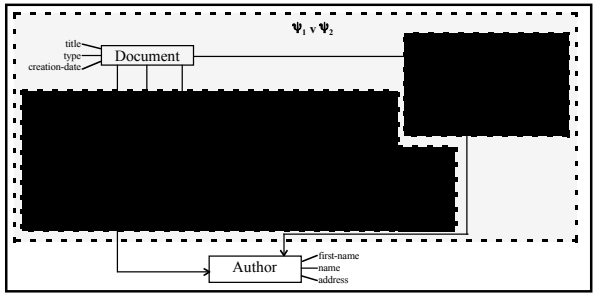

*Fig. 8. An example of an **upper limit** of two subschemas*

Definition 10 : Lower limit of two subschemas

The lower limit of two subschemas ψ_1 et ψ_2 of a schema Σ is the greatest subschema which is contained both in ψ_1 and ψ_2. Formally, the lower limit of two subschemas $\psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1})$ and $\psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2})$, denoted by $\psi_1 \wedge \psi_2$, is defined as:

$\psi_3 = \psi_1 \wedge \psi_2 = (C_{\psi_3}, \rho_{\psi_3}, \lambda_{\psi_3})$ where :

- $C_{\psi_3} = C_{\psi_1} \cap C_{\psi_2}$
- $\forall c \in C_{\psi_3}, \quad \rho_{\psi_3}(c) = \rho_{\psi_1}(c) \cap \rho_{\psi_2}(c)$
- $\forall c_1, c_2 \in C_{\psi_3} \quad \lambda_{\psi_3}(c_1, c_2) = \lambda_{\psi_1}(c_1, c_2) \cap \lambda_{\psi_2}(c_1, c_2)$

Example:

An example of a lower limit is shown in fig. 9.

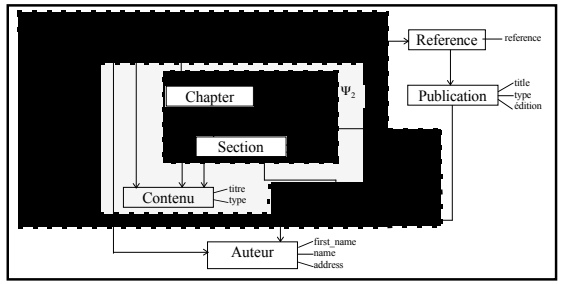

*Fig. 9. An example of a **lower limit** of two subschemas*

3. Algebraic operators

We have shown that the set of subdatabase of a database is closed under $\wedge$ and $\vee$. We now prove that this set is also closed under algebraic operators: *select, project, associate, union, intersection, difference, join, grouping, split, distribution.*

The algebraic operators have as operands a subdatabase and produce a new subdatabase. In this algebra, the closure property is maintained, because the results produced by the queries are structured in the same manner as the operands. Also, this algebra allows to produce a new class of objects that enriches the existing database.

The algebraic operators will be formally defined in the appendix. The examples used to explain these operators will make use of the database shown in fig. 1 and 3. We extend algebraic operators to subdatabases.

3.1. Select (σ)

The *Select* is a unary operator, which operates on a subdatabase to produce a new subdatabase in which objects and inter-objects links satisfy a specified predicate. The resultant subdatabase has the same schema as the operand subschema, and the instances of the result is a subset of the instances of the operands which satisfy the predicate. The select operation is denoted **σ (X) [P]** where *X* is an operand subdatabase and *P* is a predicate. The predicate *P* is a logical expression that is evaluated to *true* or *false*. This logical expression is composed by terms interconnected by logic operators (*and, or, not*). Each term can be, either a condition on an attribute value, or a condition on an inter-object link. The condition on the attribute values have the following form:

$$a\ \theta\ constant,\ or\ a\ \theta\ b$$

where *a* and *b* represent attributes.

- If *a*, *b* and *constant* are integer, then θ can be: $=, \neq, \geq, \leq, <, >$
- If *a*, *b* and *constant* are Boolean, then θ can be: $=, \neq$
- If *a*, *b* and *constant* are string, then θ can be : $=, \neq, \geq, \leq, <, >$
- If *a* is a text and *constant* is a string, then θ can be: *contains, notcontains*

To express the conditions on the cardinalities of links, we use the symbol *"→"*. The conditions on the cardinalities of links have the following form: $A \rightarrow B\ \theta\ cst$, where *A* and *B* are classes, θ is a comparison operator that can be: $=, \neq, \geq, \leq, <, >$ and *cst* is a numeric constant. For example, the predicate "*Document→Author = 2*" means that the object of the *Document* class must be linked to only two objects of the *Author* class.

Example:

*« Find the documents which are written by two authors only, whose titles contain the word '**computer**' and which are composed of **at least two chapters** ». (cf. fig. 1 and 2)*

The algebraic expression of this query is:

$X := (\{Document, Author, Chapter\}, \{(Document, Author), (Document, Chapter)\})$

$Y := \sigma(X)\ [(Document.title\ contains\ "Computer")\ and\ (Document \rightarrow Chapter > 2)\ and\ (Document \rightarrow Author = 2)]$

If we suppose that the values of the title attribute (cf fig. 2) are:

$d_1.title$ = "Computer and medicine", $d_2.title$ = "Data and Computer",

$d_3.title$ = "Data Processing", $d_4.title$ = "Computer Science"

The result of the *select* operation is shown in fig. 11.

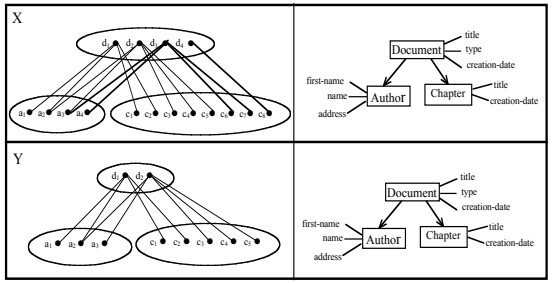

*Fig. 11. Representation of the subdatabase Y resulting from a **select** operation*

3.2. Projection (π)

The *project* is a unary operator, which operates on a subdatabase to produce a new subdatabase reduced to a subset of classes, a subset of inter-class links, and a subset of attributes. The project operation is denoted **π (X) [$c_1(p_{11}, p_{12}, ..., p_{1n})$, $c_2(p_{21}, p_{22}, ..., p_{2n})$, ..., $c_m(p_{m1}, p_{m2}, ..., p_{mn})$]**, where *X* is the subdatabase operand and the set $\{c_1, c_2, ..., c_n\}$ represents the different classes of the projected subdatabase. The set $\{p_{i1}, p_{i2}, ..., p_{in}\}$ represents the properties of the projection of the c_i class.

The project operation is valid only if the subschema $\psi_Y = (C_Y, \rho_Y, \lambda_Y)$ is such as :

- $C_Y \subset C_X$
- $\forall c \in C_Y\ \rho_Y(c) \subset (\rho_X(c) \cap C_Y)$

Example:

An example of project operation is shown in fig. 12. The expression of this project operation is:

$$Y := \pi(X)\ [Document\ (title), Author\ (first\text{-}name, name)]$$

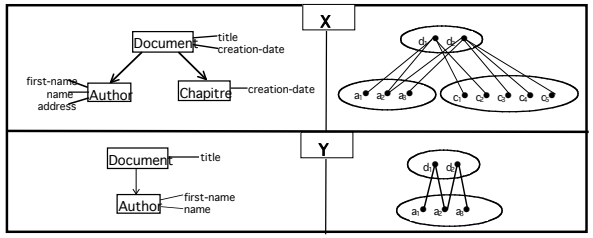

*Fig. 12. An example of a **project** operation*

3.3. Associate (↔)

The associate operator is a binary operator which constructs a new subdatabase by concatenating two subdatabases. The subschema of the result subdatabase is a concatenation of the subschemas of operands subdatabases through some inter-classes links. The instance of result subdatabase is constituted by the objects of the operands which are connected together. Since two subdatabase may have more than one link, it is necessary to specify through which links the concatenation is made. The associate operation between subdatabase *X* and *Y* through the links *L* is denoted **X↔[L]Y,** where $L=\{(c_i, c_j)\}$ is included in the set $C_X \times C_Y$.

Example:

Let *X* and *Y* be the subdatabases represented by fig. 13, an example of associate operation is:

$$Z := X \leftrightarrow [Section, Picture)]\ Y$$

The result subdatabase of this query is shown in fig. 13. In this example, the object i_1 of the *Picture* class is not represented in the result because it is not linked to any objet of the set $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ (cf. fig. 2). For the same reason, the objects $\{s_2, s_3, s_6, s_7\}$ are not present in the result, because they are not linked to any object of the set $\{i_1, i_2, i_3, i_4, i_5\}$. Consequently, the objects $\{c_2, c_5\}$ of the *Chapter* class are also omitted from the result because they are not linked to any of the objects of the *Section* class.

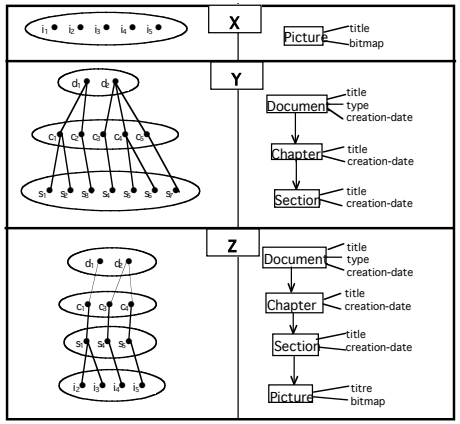

Fig. 13. An example of ***associate*** *operation (Z := X ↔ [(Section, Picture)] Y)*

3.4. Union, Intersection and Difference(+, ●, -)

The *union* operator is a binary operator which constructs a new subdatabase by combining two subdatabases. The subschema of the result subdatabase is defined as the upper limit of the subschemas of the operands subdatabases. The instance of the result subdatabase is constituted by the union of the set of instances of operands which are extended to the subschema of the result subdatabase. The union operation is denoted **X+Y.**

The *intersection* operator, as the union operator, is a binary operator which constructs a new subdatabase by combining two subdatabases. The subschema of the result subdatabase is defined as the lower limit of the subschemas of the operands subdatabases. The instance of the result subdatabase is constituted by the intersection of the set of instances of the subdatabases operands. The intersection operation is denoted **X•Y**

The *difference* operator, as the union and intersection operators, is a binary operator which constructs a new subdatabase by subtracting from a subdatabase *X* of an other subdatabase *Y*. The subschema of the result subdatabase is defined as the subschema of *X*. The instance of the result subdatabase is constituted by the instance of *X* to which is subtracted the instances of the objects of *Y*. The difference operation is denoted **X-Y**

Example:

Examples of *union*, *intersection* and *difference* operations are shown in fig. 14.

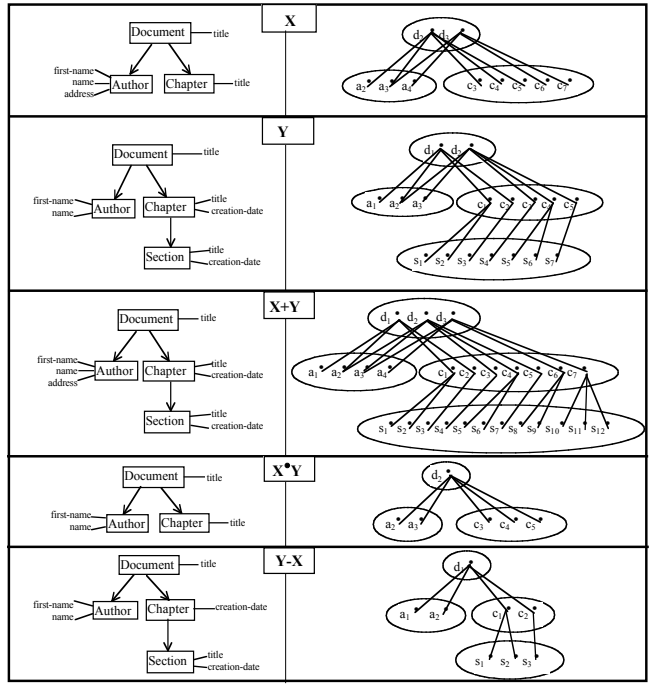

Fig. 14. Examples of ***union***, ***intersection*** *and* ***difference*** *operations*

3. 5. Join

Unlike the relationnel data model, in object-oriented models many relationships between objects can be represented within the objects themselves. Hence the join operations are used less frequently in the object algebra than in the relationnel algebra. However, the existing objects in a database may not explicitly reflect all relationships required by the queries. An explicit join is still needed to handle the cases when the relationship being queried upon is not defined within the object classes. Such relationship is called *values-based* relationship in contrast with those specified explicitly in the object model. For example, suppose we have a *layer document* class and a *technical document* class. A query « Find the layer document which has the same key words as the technical document », requires a value-based join between the classes *layer document* and *technical document* through the attribute *key-words*.

The join condition links the properties of only one class of *X* and the properties of only one class in *Y*. The subschema of the result subdatabase is defined as the concatenation of the subschemas of the subdatabases operands through a new class *c*. This new class is defined as a set of tuples. The tuple is made of two attributes, the first attribute is a set of references to the class c_1 of *X*, the second attribute is a set of references to a class c_2 of *Y*. The graphical structure of class *c* is shown in fig 15. The instance of the result subdatabase is constituted of a portion of *X* that satisfies the join condition, a portion of *Y* that satisfies the join condition, and the union of the instances of

operands which are extended to the subschema of the result subdatabase. The Join operation is denoted by **X ⊗ [P] Y** where *X* and *Y* are the subdatabases operands and *P* represents the join predicate.

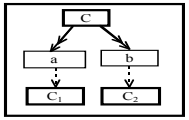

*Fig 15. The graphic representation of the structure of the class **C***

Example:
Let *X* and *Y* be the subdatabases represented by the fig. 16. An example of join operation is :

$$X \otimes [\ Chapter.date\text{-}creation = Part.date\text{-}creation\ ]\ Y$$

if, we suppose that the values of the attributes *date-creation* of the *reference-guide* class and of the *user-guide* class are:

$c_1.creation\text{-}date$ = "26-03-62", $c_2.creation\text{-}date$ ="26-03-62",
$c_3.creation\text{-}date$ = "05-07-62", $c_4.creation\text{-}date$ = "19-03-62",
$p_1.creation\text{-}date$ = "26-03-62", $p_2.creation\text{-}date$ ="26-03-62",
$p_3.creation\text{-}date$ ="05-07-62", $p_4.creation\text{-}date$ = "05-07-62".

The result of the *join* operation is shown in fig. 16. The subschema of the result subdatabase is composed of the subschema of *X* and of *Y*, and of a new *Chapter-Part* class which refer to *Chapter* class and *Part* Class. The *Chapter-Part* class is composed of two objects f_1 and f_2. The object f_1 is linked to the objects $\{c_1, c_2\}$ of the *Chapter* class and it is also linked to the objects $\{p_1, p_2\}$ because the two sets of objects have the same value of the *date-creation* attribute and the *join* condition is fulfilled. For the same raison, the object f_2 is linked to the objects $\{p_3, p_4\}$. However, the object c_4 of *Chapter* class is not present in the result, because it doesn't fulfill the *join* condition. consequently, the objects $\{s_7, s_8\}$ of the *Selection* class are also omitted from the result.

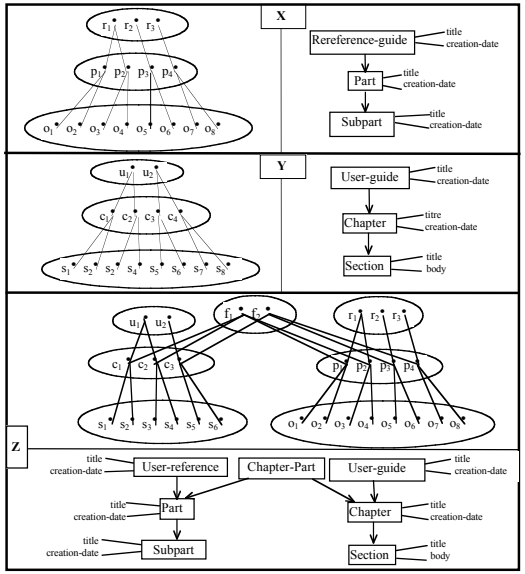

*Fig. 16 Representation of the subdatabases **X, Y** and **Z***

3.6 Grouping (γ)

The *grouping* is an unary operator. This operator is also used in NST-Algebra [Guti89]. It is used to transform an association relationship to an aggregation relationship. The grouping operation is defined as **γ(X)[c₁, c₂, a]** where *X* is an operand subdatabase and $[c_1, c_2, a]$ is the association relationship to be transformed to an aggregation relationship.

Example:
An example of a grouping operation is shown in fig 17.

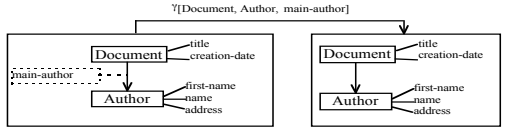

*Fig. 17 Example of the **grouping** operation*

3.7. Split (ε)

The *split* operation is the inverse of the *grouping* operation. Indeed, the *split* operation is used to transform an aggregation relationship to an association relationship. The split operation is defined as **ε(X)[c₁, c₂, a]** where *X* is an operand subdatabase and $[c_1, c_2, a]$ is the aggregation relationship to be transformed to an association relationship**.**

Example:
An example of the *split* operation is shown in fig 18.

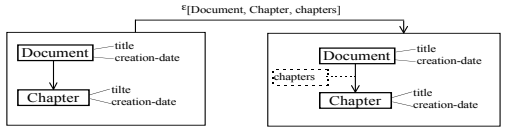

*Fig. 18 Example of the **split** operation*

3.8 Distribution (δ)

The *distribution* operator is used to restructure a subdatabase. This restructuration is made by distributing a property of class c_1 to a class c_2. The distribution is valid, if there exists a link between c_1 and c_2. The distribution operation is defined as **δ(X)[c₁, c₂, p]** where X is a subdatabase operand, p is the property of c_1 to be distributed to class c_2.

Example:

An example of the *distribution* operation is shown in figure 19.

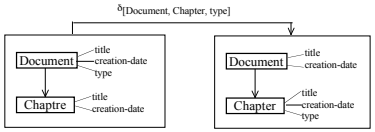

*Fig. 19 Example of the **distribution** operation*

4. Examples of expressions of algebraic queries

A storage and a query system of structured documents using the OODBMS technology was proposed in [Chri94]. This system was developed in the Verso project of INRIA. It is based on an extension of the model and query language of O2 OODBMS [Odeu90]. In order to request documents by their structure and content, *O₂SQL* was extended to:

- take into account new functions of modeling like the ordered n-tuples and unions of types
- include predicates of textual retrieval like contains.

We will present examples of queries to compare our algebraic approach with the *O₂SQL* query language.

Example 1:

*«Find the documents containing the word **Computer** in the **title** and which are constituted of **two chapters**.»*

The request is written as follows in the *O₂SQL* query language:

Select *x*
From *x **in** Document*
*y **in** x.Chapter*
Where *x.title **contains** « computer » and count (x.chapter) =2*

The algebraic expressions of the request are :

R_1 := **π** *[Document, Chapter] ((Document.title **contains** « computer »)*
∧ (Document→Chapter = 2))
R_2 := **π** *[R_1] (Document)*

R_1 := **σ** *[Document, Chapter] (Document.title **contains** « computer »)*
R_2 := **σ** *[R_1] (Document→Chapter = 2)*

This example represents an interrogation on the content of the document as well as its structure. The formulation of such a query under an algebraic form allows a uniform specification of the condition on the content and on the structure. However, the interrogation, using *O₂SQL*, can only be made by using key words such as: *Count*.

Example 2:

*« Find the documents that are **only** constituted of **two chapters** with the following condition: each **chapter** belonging to **three documents**.».*

The algebraic expression of the query is :

R_1 := **σ** *[Document, Chapter] (Document→Chapter = 2)*
R_2 := **σ** *[Document, Chapter] (Chapter→Document = 3)*
$R_3 := R_1 \bullet R_2$
R_4 := **π** *[R_3] (Document)*

To our knowledge, this query cannot be expressed in *O₂SQL*. Indeed, such a query expresses a condition on the structure starting from the document to chapters and inversely, from chapters to the document.

5. Conclusion

We have presented in this paper an algebra for structured documents in an object-oriented approach. We have defined the concept of subdatabase on which is based the definition of algebraic operators. Indeed, the operands and the query result are subdatabases, as a consequence, the closing condition and the operators orthogonality conditions are respected because the operands and the results of the query operators are structured in the same way. The operators can be regrouped into two categories. A first category that deals with information retrieval operators (select, project, associate, union, intersection, difference and join operators). A second category that deals with information restructurating operators (grouping, split, distribution operators).

Appendix

The formal definition of the algebraic operators are given below :

- Select

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$. The *select* operation is defined as:

$Y = \sigma(X)[P] = (\psi_X, O_Y, A_Y)$ *where* $\psi_Y = (C_Y, \rho_Y, \lambda_Y)$ such as :

- $C_Y = C_X$,
- $\forall c \in C_Y, \quad \rho_Y(c) = \rho_X(c)$,
- $\forall c_1, c_2 \in C_Y, \quad \lambda_Y(c_1, c_2) = \lambda_X(c_1, c_2)$
- $O_Y = \{o / o \in O_X : P(o) = true\}$
- $A_Y = \{(x, y) / (x, y) \in A_X : P(x) = true \text{ and } P(y) = true\}$

- Project

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$. The *project* operation is defined as:

$Y = \pi(X)[c_1, c_2, ..., c_n] = (\psi_X, O_Y, A_Y)$ *where* $\psi_Y = (C_Y, \rho_Y, \lambda_Y)$ such as:

- $C_Y = \{c_1, c_2, ..., c_n\}$
- $\forall c \in C_Y \quad \rho_Y(c) = \rho_X(c) \cap C_Y$
- $\forall c_1, c_2 \in C_Y, \quad \lambda_Y(c_1, c_2) \subseteq \lambda_X(c_1, c_2)$
- $O_Y = \{o / o \in O_X : \tau(o) \in C_Y\}$
- $A_Y = \{(x, y) / (x, y) \in A_X : \tau(y) \in \rho_Y(\tau(x))\}$

- Associate

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$, $Y = (\psi_Y, O_Y, A_Y)$ $(\psi_Y = (C_Y, \rho_Y, \lambda_Y))$ and a database $D = (\Sigma, O, A)$. The *association* operation is defined as:

$Z = X <\text{-->} [L]Y = (\psi_Z, O_Z, A_Z)$ *where* $\psi_Z = (C_Z, \rho_Z, \lambda_Z)$ such as:

- $C_Z = C_X \cup C_Y$
- $\forall c \in C_Z, \quad \rho_Z(c) = \rho_X(c) \cup \rho_Y(c) \cup \{c'/(c,c') \in L\}$
- $\forall c_i, c_j \in C_Z \begin{cases} \text{if } (c_i,c_j) \in (C_X \times C_X) \text{ then } \lambda_Z(c_i,c_j) = \lambda_X(c_i,c_j) \\ \text{if } (c_i,c_j) \in (C_Y \times C_Y) \text{ then } \lambda_Z(c_i,c_j) = \lambda_Y(c_i,c_j) \end{cases}$
- $O_Z = \{ o / o \in (O_X \cup O_Y) : \forall (o,o') \in ((O_X \cup O_Y) \times (O_X \cup O_Y)) \text{ such that :}$
  $\text{if } (\tau(o),\tau(o')) \in L \text{ then } (o,o') \in A \text{ or if } \tau(o') \in \rho_Z(\tau(o)) \text{ then } (o,o') \in (A_X \cup A_Y) \}$
- $A_Z = \{(x,y) / (x,y) \in A_X : x \in O_Z \text{ and } y \in O_Z\} \cup \{(x,y) / (x,y) \in A_Y : x \in O_Z \text{ and } y \in O_Z\} \cup$
  $\{(x,y) / (x,y) \in A : (\tau(x),\tau(y)) \in L \text{ and } x \in O_Z \text{ and } y \in O_Z\}$

- Union

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$ and $Y = (\psi_Y, O_Y, A_Y)$ $(\psi_Y = (C_Y, \rho_Y, \lambda_Y))$. The *union* operation is defined as:

$Z = X + Y = (\psi_Z, O_Z, A_Z)$ *where* $\psi_Z = (C_Z, \rho_Z, \lambda_Z)$ such that :

- $\psi_Z = \psi_X \vee \psi_Y$
- Let $\overline{X}$ such as: if $\psi_X \neq \psi_Z$, then $\overline{X} = (\psi_{\overline{X}}, O_{\overline{X}}, A_{\overline{X}})$ such that $\psi_{\overline{X}} = \psi_Z$ and $\pi(\overline{X})(C_X) = X$
  else $\overline{X} = X$
- Let $\overline{Y}$ such as: if $\psi_Y \neq \psi_Z$, then $\overline{Y} = (\psi_{\overline{Y}}, O_{\overline{Y}}, A_{\overline{Y}})$ such that $\psi_{\overline{Y}} = \psi_Z$ and $\pi(\overline{Y})(C_Y) = Y$
  else $\overline{Y} = Y$
- $O_Z = O_{\overline{X}} \cup O_{\overline{Y}}$
- $A_Z = A_{\overline{X}} \cup A_{\overline{Y}}$

- Intersection

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$ and $Y = (\psi_Y, O_Y, A_Y)$ $(\psi_Y = (C_Y, \rho_Y, \lambda_Y))$. The *intersection* operation is defined as:

$Z = X \bullet Y = (\psi_Z, O_Z, A_Z)$ *where* $\psi_Z = (C_Z, \rho_Z, \lambda_Z)$ such as:

- $\psi_Z = \psi_X \wedge \psi_Y$
- $O_Z = O_X \cap O_Y$
- $A_Z = A_X \cap A_Y$

- Difference

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$ and $Y = (\psi_Y, O_Y, A_Y)$ $(\psi_Y = (C_Y, \rho_Y, \lambda_Y))$. The *difference* operation is defined as:

$Z = X - Y = (\psi_Z, O_Z, A_Z)$ *where* $\psi_Z = (C_Z, \rho_Z, \lambda_Z)$ such as:

- $\psi_Z = \psi_X$
- $O_Z = O_X - O_Y$
- $A_Z = A_X - A_Y$

- Join

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$ and $Y = (\psi_Y, O_Y, A_Y)$ $(\psi_Y = (C_Y, \rho_Y, \lambda_Y))$. The *join* operation is defined as:

$Z = X \otimes [P(c_1, c_2)]Y = (\psi_Z, O_Z, A_Z)$ *where* $\psi_Z = (C_Z, \rho_Z, \lambda_Z)$ such as :

- $C_Z = C_X \cup C_Y \cup \{c\}$, where c is the new class created by the join operation
- $\forall x \in C_Z \begin{cases} \text{if } x \in C_X, \text{ then } \rho_Z(x) = \rho_X(x) \\ \text{if } x \in C_Y, \text{ then } \rho_Z(x) = \rho_Y(x) \\ \text{if } x = c, \text{ then } \rho_Z(x) = \{c_1, c_2\} \end{cases}$
- $\forall c_i, c_j \in C_Z \begin{cases} \text{if } (c_i,c_j) \in (C_X \times C_X) \text{ then } \lambda_Z(c_i,c_j) = \lambda_X(c_i,c_j) \\ \text{if } (c_i,c_j) \in (C_Y \times C_Y) \text{ then } \lambda_Z(c_i,c_j) = \lambda_Y(c_i,c_j) \end{cases}$
- $O_Z = \{o / o \in (O_X \cup O_Y) : (\text{if } \tau(o) = c_1 \text{ then } P(o) = True) \text{ and } (\text{if } \tau(o) = c_2 \text{ then } P(o) = True)\}$
- $A_Z = \{(x,y) / (x,y) \in (A_X \cup A_Y) : P(x) = True \text{ and } P(y) = True\} \cup$
  $\{(x,y) / \tau(y) = c_1 \text{ and } \tau(x) = c \text{ and } P(y) = True\} \cup$
  $\{(x,y) / \tau(y) = c_2 \text{ and } \tau(x) = c \text{ and } P(y) = True\}$

- Grouping

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$. The *grouping* operation is defined as:

$Y = \gamma(X)[c_1, c_2, a] = (\psi_Y, O_Y, A_Y)$ *where* $\psi_Y = (C_Y, \rho_Y, \lambda_Y)$ such as:

- $C_Y = C_X$
- $\forall c \in C_Y, \quad \rho_Y(c) = \rho_X(c)$
- $\forall c_i, c_j \in C_Y \begin{cases} \text{if } c_i \neq c_1 \text{ and } c_j \neq c_2 \text{ then } \lambda_Y(c_1,c_2) = \lambda_X(c_1,c_2) \\ \text{else } \lambda_Y(c_1,c_2) = \lambda_X(c_1,c_2) - \{a\} \end{cases}$
- $O_Y = O_X$
- $A_Y = A_X$

- Split

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$. The *split* operation is defined as:

$Y = \varepsilon(X)[c_1, c_2, a] = (\psi_Y, O_Y, A_Y)$ *where* $\psi_Y = (C_Y, \rho_Y, \lambda_Y)$ such as:

- $C_Y = C_X$
- $\forall c \in C_Y, \quad \rho_Y(c) = \rho_X(c)$
- $\forall c_i, c_j \in C_Y \begin{cases} \text{if } c_i \neq c_1 \text{ and } c_j \neq c_2 \text{ then } \lambda_Y(c_1,c_2) = \lambda_X(c_1,c_2) \\ \text{else } \lambda_Y(c_1,c_2) = \lambda_X(c_1,c_2) \cup \{a\} \end{cases}$
- $O_Y = O_X$
- $A_Y = A_X$

- Distribution

Let us consider a subdatabase $X = (\psi_X, O_X, A_X)$ $(\psi_X = (C_X, \rho_X, \lambda_X))$. The *distribution* operation is defined as:

$Y = \delta(X)[c_1, c_2, p_{1i}] = (\psi_Y, O_Y, A_Y)$ *where* $\psi_Y = (C_Y, \rho_Y, \lambda_Y)$ such as:

- $C_Y = C_X - \{ c_1(p_{11}, p_{12}, \ldots, p_{1(i-1)}, p_{1i}, p_{1(i+1)}, \ldots, p_{1n}), c_2(p_{21}, p_{22}, \ldots, p_{2m}) \} \cup$
  $\{ c_1(p_{11}, p_{12}, \ldots, p_{1(i-1)}, p_{1(i+1)}, \ldots, p_n), c_2(p_{21}, p_{22}, \ldots, p_{2m}, p_{1i}) \}$
- $\forall c \in C_Y, \quad \rho_Y(c) = \rho_X(c)$
- $\forall c_i, c_j \in C_Y, \quad \lambda_Y(c_i,c_j) = \lambda_X(c_i,c_j)$
- $O_Y = O_X$
- $A_Y = A_X$

References

[Alha93] Alhajj, R. and Arkun, M.E. "An Object Algebra For Object-Oriented Database Systems". Database, 1993, Vol. 24, N°3, p. 13-22

[Amgh89] Amghar, Y. « Base d'Objets Documentaires Modélisation-Manipulation-Stockage de Documents Codés Selon ODA », Ph D. these, INSA de Lyon, 1989, 189 p.

[Bens89] Bensadoun, O. & Chrisment, C. & Pujolle, G. and Zurfluh, G. "Aspects Dynamiques dans les bases de documents". Actes de 5éme Journées Base de Données Avancées, Geneve (Swizerland), September 26-28, 1989, p. 291-308.

[Clue90] Cluet, S. & Delobel, C. & Lecluse, C. and Richard, P. "RELOOP: An Algebra Based Query Language For Object-Oriented Database System". Data and Knowledge Enginnering, 1990, Vol. 5, p. 333-351.

[Chri94] Christophides, V. & Abiteboul, S. & Cluet, S. And Scholl, M. « From Structured Documents to Novel Query Facilities ». SIGMOD'94, Minneapolis, May 1994, p.313-324.

[Flor82] Flory, A. and Metzger, J.O. " Exemples d'application du modèle relationnel à des bases textuelles". Inforsid 82, Toulouse, Mai, 1982, 23 p.

[Guo93] Guo, M. "An object-oriented SQL(OSQL) Based on Association Patten Query Formulation". In Proceedings of the Phecnix Conference On Computers and Communications, Tempe (California), March, 1993, p. 231-237.

[Guti89] Güting, R.H. & Zicari, R. and Choy, D.M. "An Algebra for Structured Office Documents". ACM Transaction on Office Information Systems, 1989, Vol. 7, N° 2, p. 123-157.

[Hamz95] Hamza, H "An Object Algebra Based on Relationships Between Objects". In Proceedings of the Tenth International Symposium on Computer and Information Sciences, Ephesus (Izmir, Turkey), October 30-November 1, 1995, 305-312.

[Hamz96] Hamza, H. « An algebra for structured documents in the context of object-oriented approach », Ph D. these, INSA de Lyon, 1996, 200 p.

[Liu93] Liu, L. "A Recursive Object Algebra Based on aggregation abstract For manipulating Complex Objects". Data and Knowledge Engineering, August, 1993, Vol. 11, N° 1, p. 21-60.

[Macl91] Macleod, I. & Narnard, D. & Hamilton, D. and Levison, M. "SGML Documents and Non-Lnear Text Retrieval". RIAO'91, Barcelona (Spain), April 2-5, 1991, p.226-244.

[Mira84] Miranda, S. "Système Documentaire et SGBD Relationnel l'approche MINIDOC". INFORSID84, May, 1984, Bandol, p. 129-148.

[Odeu90] O-Deux et Al « The story of O2 ». IEEE Transaction On Knowledge and Data Engineering, 1990, Vol. 2, N°1, p. 91-108.

[Osbo88] Osborn, S. "Identity, Equality and Query Optimization". Advances in Object-Oriented Database, 2nd International Workshop on Object-Oriented Database System, Bad Münster am (Germany), September 27-30, 1988, p. 346-351.

[Schw96] Schwer, S.R. « Trees in informations Systems ». Internal report, CRI 1997, Sorbonne university ParisI.

[Shaw90] Shaw, G.M. and Zdonik, S.B. "A Query Algebra For Object-Oriented Databases". In Proceedings of the International Conference on Data Engineering, Los Angeles (California), February 5-9, 1990, p. 154-162.

[Ston88] Stonebraker, M. & Stettner, H. & Lynn, N. & Kalash, J. and Guttman, A. "Document Processing in a Relational Database System". ACM Transactions on Office Information Systems, 1988, Vol. 1, N°2, p. 143-158.

[Su93] Su, S.Y.W. & Guo, M. and Lam, H."Association Algebra: A Mathematical Foundation for Object-Oriented Databases". IEEE Transactions on Knowledge and Data Engineering, 1993, Vol. 5, N°5, p. 775-798.

[Subr95] Subramanian, B. & Leung, T.W. & Vandenberg, S.L. and Zdonik, S.B. "The AQUA Approch to Querying Lists and Trees in Object-Oriented Databases". In Proceedings of the International Conference on Data Engineering, Taipei (Taiwan), March 6-10, 1995, p. 80-89.

[Vand91] Vandenberg, S.L. and Dewitt D.J. "Algebraic Support for Complex Objects with Arrays, Identity, and Inheritance". In Proceedings of the International Conference on Management of Data, Denver (Colorado), May 29-31, 1991, p. 158-167.